Unbreakable DevOps on Red Hat OpenShift

Peter Hack
Technology Strategist

Florian Bacher
Technology Strategist
Intro – today’s world

What is the Challenge to Unbreakable DevOps?
Digital transformation is software defined

**The way we fly**
Automation is a key priority for airlines from check in, to identification, to boarding.

**The way we transact**
50% use smart devices to access financial services.

**The way trains are managed**
BNSF is using IoT to detect imminent bearing failures before they happen.

**The way we get around**
BMW and other car manufacturers accelerating application experiences.

**The way we shop**
Driverless supermarket shopping delivery with Kroger.

**The way we play**
Carnival cruises introduces IoT enabled payment using a smartwatch.
And the software driving the business needs to work \textbf{P E R F E C T L Y}
Our own journey

We have done this and so can you
Who is Dynatrace?

9x
Gartner Magic Quadrant leader 9 years in a row

#1
Market Share (Gartner, IDC)

750%
YoY Growth of the new all-in-one platform

26
Major releases per year

72 / 100
Customers in Fortune 100
Six and half years ago...

Dynatrace journey to NoOps begins
Vision of “self-driving” DevOps ...
Achieving, seeing, sharing success!

- Releases per year: 2 in 2011, 26 in 2017
- Production bugs reported by customers: 100% in 2011, 7% in 2017
- EC2 instances: 5 in 2011, ~1000 in 2017
- Daily deployments: 0 in 2011, 5000 in 2017
Collecting more evidence: https://dynatrace.ai/acsurvey

Commit Cycle Time: From Dev to Pro

- 95th Percentile
- Median
- Goal: 1h to Production

- Small (11-100 employees)
- Medium (101-1000 employees)
- Large (1001-5000 employees)
- Extra large (over 5000 employees)
Verdict: The Majority is not delivering high quality faster

<table>
<thead>
<tr>
<th>Metric</th>
<th>95th Percentile</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to Production (Commit Cycle Time)</td>
<td>2 days</td>
<td>12.5 days</td>
</tr>
<tr>
<td>Business Impacting Deployments</td>
<td>1 out of 10</td>
<td>3 out of 10</td>
</tr>
<tr>
<td>Per Production Deployment</td>
<td>0 hotfixes</td>
<td>3 hotfixes</td>
</tr>
<tr>
<td>MTTR (Mean Time to Repair)</td>
<td>~4 hours</td>
<td>4.8 days</td>
</tr>
</tbody>
</table>

Evaluate for yourself: https://dynatrace.ai/acsurvey
How does an organization from Median to 95th percentile!

**Faster**

97% deployment lead time

**More Frequently**

12 to 26 releases per year

**Better**

75% production incidents

- **Automated Quality (Shift-Left)**
  - Goal: Stop Bad Code Changes Early & Automated
  - Automated Testing & Quality Gates

- **Automated Multi-Stage Deployments (Shift-Right)**
  - Goal: Increase deployments into stable environments
  - Dark, Shadow, Blue/Green Deployments with Auto-Validation

- **Automated Operations (Self-Healing)**
  - Goal: NoOps & Zero-Impact on End Users
  - Automated Remediation & (On-Demand) Scaling
The DevOps journey has a world-class monitoring roadmap

- Chaotic Descriptive
  - Why did it happen?

- Reactive Predictive
  - What will happen?

- Servicing Prescriptive
  - How can we prevent it from happening?

- Value Adaptive
  - How can we automate processes so we can innovate faster?

Visibility & Intelligence

Individual component ➔ full-breadth ➔ correlation ➔ as-a-service ➔ AIOps
The DevOps journey is built on a solid foundation

- **Multi-Platform Monitoring**
  - Automated monitoring – monitoring as feature of the end-to-end pipeline

- **Quality Shift-Left**
  - Automate quality (shift-left) – automate the pipeline and stop bad code changes before they reach prod

- **Deployment Shift-Right**
  - Automate deployment (shift-right) – push “monitoring-as-code” for auto-validation and auto-alerting

- **Self Healing**
  - Automate operations (self-healing) – auto-mitigate bad deployments in production
delivering better software faster

“Any journey begins with the first step”
The first step
Is the order correct? 

Yes

no

Ingredients? 

Correct

Yes

Cooked Correctly? 

Yes

no

All Items Prepared? 

Yes

no

Feedback

Requirements → Checkout → Build → Packaging → Delivery
What we want

commit  build  test  stage  prod
✓  ✓  ✓  ✓  ✓

Automatically push to next stage (quality gates)

What we (sometimes) have

commit  build  test  stage  pre-prod1  pre-prod2  prod
✓  ✓  ✗  ✓  ✗  ✓  ✗
Unbreakable delivery pipeline in action

1. Staging
   - Push Context
   - CI/CD

2. Approve Staging
   - Auto-Quality Gate
   - CI/CD

3. Production
   - Push Context
   - CI/CD

4. Approve Production
   - Auto-Validate
   - CI/CD

5. Production
   - Auto-Remediate!
   - CI/CD

Build #17
Build #18
Automated Deployment Validation

https://github.com/keptn/pitometer

https://github.com/keptn,
www.keptn.sh
Automate Architectural Checks into CI/CD/CO!

Identify / Optimize Architectural Patterns
Recursive Calls, N+1 Call Pattern, Chatty Interfaces, No Caching Layer ...
How is Performance & Resource Consumption per Service Endpoint?
From Google: “Everything as Code” e.g. Enforce Architectural Rules
From Dynatrace: “Performance Signature as Code” evaluated through Jenkins

“Performance Signature” for Build Nov 16

“Performance Signature” for Build Nov 17

“Multiple Metrics” compared to prev Timeframe

Simple Regression Detection per Metric

“Performance Signature” for every Build
Pitometer (part of @keptnProject): Metrics-based grading of a Deployment!

**Pitometer Specfile**

```json
{
  "spec_version": "1.0",
  "indicators": [
    {
      "id": "gp_persistent_alloc_bytes",
      "source": "Prometheus",
      "query": "avg(go_gc_duration_seconds{instance=localhost:9090})",
      "grading": {
        "type": "threshold",
        "thresholds": {
          "upperSevere": 2180000000000,
          "upperWarning": 2000000000000
        }
      },
      "metricScore": 20
    },
    {
      "id": "conversion_rate",
      "source": "Dynatrace",
      "query": {
        "timeSeriesId": "com.dynatrace.builtin:app.conversionrate",
        "aggregation": "avg",
        "tags": {
          "SockShop-Blue"
        }
      },
      "grading": {
        "type": "threshold",
        "thresholds": {
          "lowerSevere": 0.05,
          "lowerWarning": 0.02
        }
      },
      "metricScore": 20
    },
    {
      "objectives": {
        "pass": 10,
        "warning": 15
      }
    }
  ]
}
```

**Source**

- > 2GB: 0 Points
- < 2GB: 20 Points

**Grader**

- > 5%: 20 Points
- < 5%: 10 Points
- < 2%: 0 Points
Pitometer: Dynatrace Smartscape Data Source

Vertical Dependencies

Horizontal Dependencies

fromRelationships.calls  toRelationships.calls
Pitometer: Dynatrace Smartscape Data Source Example

```json
{
  "id": "OutgoingDependencies_Backend",
  "source": "Dynatrace",
  "query": {
    "entityType": "SERVICE",
    "smartscape": "toRelationships:calls",
    "aggregation": "count",
    "tags": ["service:cart"]
  },
  "grading": {
    "type": "Threshold",
    "IgnoreEmpty": true,
    "thresholds": {
      "upperSevere": 2,
      "lowerSevere": 0
    },
    "metricScore": 10
  }
}
```
Pitometer: Run Standalone - https://github.com/keptn/pitometer

```javascript
const Pitometer = require('@pitometer/pitometer').Pitometer;
const DynatraceSource = require('@pitometer/source-dynatrace').Source;
const PrometheusSource = require('@pitometer/source-prometheus').Source;
const ThresholdGrader = require('@pitometer/grader-threshold').Grader;
const pitometer = new Pitometer();

// Register a Prometheus source that will be used if the source ID in your
// perfspec matches 'Prometheus'
pitometer.addSource('Prometheus', new PrometheusSource({
  queryUrl: '<PROMETHEUS_PROXY_ENDPOINT>',
}));

// Register a source that will be used if the source ID in your perfspec matches
// 'Dynatrace'
pitometer.addSource('Dynatrace', new DynatraceSource({
  baseURL: '<DYNATRACE_ENVIRONMENT_URL>',
  apiToken: '<DYNATRACE_API_TOKEN>',
  // Optional: A logger to be used for debugging API requests
  // log: console.log,
}));

// Register a grader for thresholds that will be used if the grader type
// matches 'Threshold'
pitometer.addGrader('Threshold', new ThresholdGrader());

// Load a perfspec - see the samples directory
const perfspec = require('./samples/monspec-sample.json');

// Run the perfspec, appening in an optional context parameter 'prod'
// and log the result out to the console
pitometer.run(perfSpec, 'prod')
  .then((results) => console.log(JSON.stringify(results)))
  .catch((err) => console.error(err));
```

```
{ "totalScore": 10,
  "objectives": {
    "pass": 30,
    "warning": 15
  },
  "indicatorResults": [
    {
      "id": "go_memstats_alloc_bytes",
      "score": 0,
      "violations": [
        {
          "value": 31808321,
          "key": "localhost:8080",
          "breach": "upper_critical",
          "comparison": "fixed",
          "threshold": 21802344
        }
      ]
    },
    {
      "id": "ConversationRate",
      "score": 10,
      "violations": [
        {
          "value": 3.9,
          "key": "sockshop-blue",
          "breach": "lower_warning",
          "comparison": "fixed",
          "threshold": 5
        }
      ]
    }
  ]
```
Autonomous Cloud Control Plane

1: push
2: deploy (shadow)
3: test (performance)
4: evaluate (scalability KPIs)
5: promote
6: deploy (blue/green)
7: evaluate (business KPIs)
8: operate (NoOps)
keptn.sh - OpenSource framework for unbreakable pipeline and more ...

**Enterprise-grade framework for shipping and running cloud-native applications**

Deployable on any Kubernetes cluster, kepntn converts any Kubernetes cluster into a self-healing, autonomous cloud fabric.

**CORE CAPABILITIES**

- Automated multi-stage unbreakable delivery pipelines
- Self-healing blue / green deployments
- Event-driven runbook automation

**DESIGN PRINCIPALS**

- GitOps-based collaboration
- Operator patterns for all logic components
- Monitoring-and-operations-as-code

- Built-on and for Kubernetes
- Event-driven and serverless
- Pluggable tooling
Resources

• Keptn & Pitometer
  • www.keptn.sh
  • github.com/keptn
  • github.com/keptn/pitometer

• Performance, Resiliency & Availability Content
  • Adrian Hornsby (AWS): https://speakerdeck.com/adhorn/resiliency-and-availability-design-patterns-3742b5ba-e013-4f50-8512-00a65775f478
  • Acacio Cruz (Google): https://www.spreaker.com/user/pureperformance/066-load-shedding-sre-at-google-with-aca
  • Thomas Steinmaurer (Dynatrace): https://www.neotys.com/performance-advisory-council/thomas_steinmaurer
Demo Time
Dynatrace AI-powered better data makes automation & unbreakable CD possible

High fidelity data  Mapped end-to-end  Deterministic AI  Answers + Action

Automated problem detection
Business impact determined
Root cause explained
No alert storms
Trigger self healing

Completely automated
Sample Dynatrace global DevOps customers
Speaking Sessions (recap)

- **The machines are coming, but you have nothing to fear**
  - Day/Time: Tuesday, May 7, 9:30 a.m.-10:15 a.m.
  - Speaker: Dave Anderson

- **Building autonomous operations for Kubernetes with keptn**
  - Day/Time: Tuesday, May 7, 2:30 p.m.-3:15 p.m.
  - Speaker: Alois Reitbauer

- **Shift-left site reliability engineering for self-healing applications**
  - Day/Time: Wednesday, May 8, 10:30 a.m.-11:15 a.m.
  - Speaker: Jürgen Etzlstorfer

- **Unifying OpenShift cluster, container and application monitoring**
  - Day/Time: Wednesday, May 8, 3:30 p.m.-3:50 p.m.
  - Speaker: Asad Ali

- **Unbreakable DevOps on Red Hat OpenShift**
  - Day/Time: Thursday, May 9, 1:00 p.m.-1:45 p.m.
  - Speaker: Peter Hack

- Full speaking session information available [HERE](#)
Stay engaged with us!

• Download a free copy of the Gartner MQ
• Take our survey https://dynatrace.ai/acsurvey
• Catch our PurePerformance Podcast Series
  https://www.spreaker.com/user/pureperformance
• Check out our Performance Clinics https://bit.ly/2pzXXIK
• Try our free trial! https://www.dynatrace.com/trial/